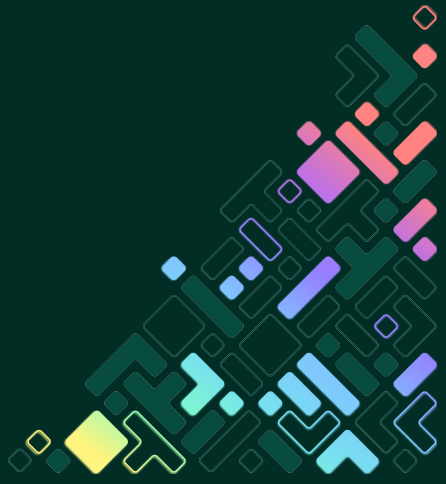
EMBEDDED
LINUX
CONFERENCE

AMP Virtio

A new Virtio transport for AMP systems

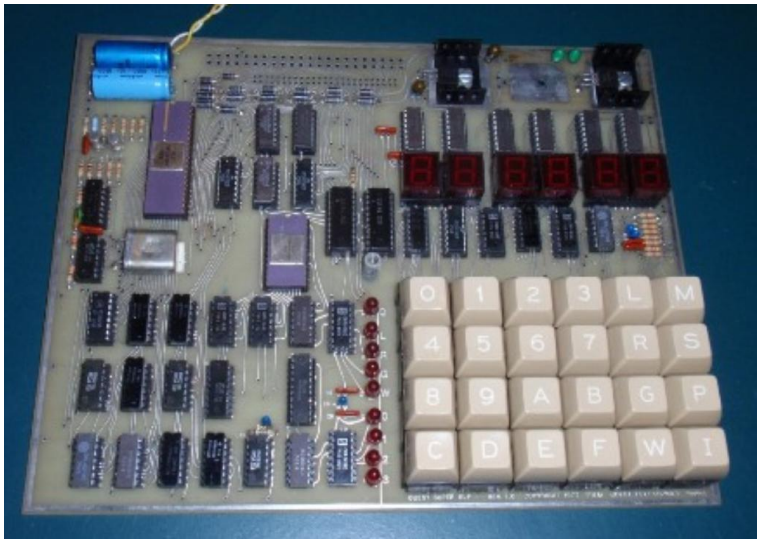


bill.mills@linaro.org

Who am I / we?

- Bill Mills
  - Linaro for 3.5 years
  - Member of OpenAMP for ~7 years
  - Founding member of the Yocto Project
  - Focused on Open Source since 2008
  - Professionally programing in the embedded world since 1986
  - “Programmer” since 1978
- Linaro
  - Nexus of Open Source for Arm ecosystem
  - Many missions but one is:
    - **V**irtio for EDGE
- OpenAMP
  - Community Project to make AMP systems more standard and easier to use
  - RTOS Portable libraries for RPMSG and adding other Virtio

Photo credit: old computer museum



My first computer, Circa 1978
1.79 MHz & 256 Bytes of RAM

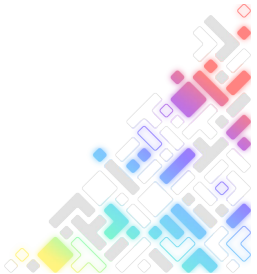
EMBEDDED OPEN SOURCE SUMMIT

Opinions and positions presented here are my own and do not necessarily reflect my employer or the projects I mention.

What am I talking about?

- ELC track but also touches Zephyr, Real Time, and Functional Safety

- Background on what we are doing and why
- Review of virtio-mmio
- Problems and Solutions for mmio on AMP
- Status and Code Pointers
- A Probable Change in Direction
- Wrap-up



Why Virtio?

Virtio

- Already supported by many OS'es, bootloaders, and hypervisors
- New projects can pick one PV device interface on which to focus
- The one to choose when you can only afford to choose one.

AMP Virtio

- Traditionally virtio is used in a hypervisor environment
- With AMP virtio we want to make virtio work where it can't today
- At a prototypical level it will work anywhere you have a **shared memory** and **bi-direction notification**
- Has uses even when memory sharing is not constrained

Why Xen and Zephyr?

Xen

- A great reference for any Type 1 hypervisor
- A type 1 hypervisor is important for functional safety
- Xen can deliver Real-time to one set of physical CPUs and let others be best effort and more flexible
- Long standing Open Source project with open governance
- Interest and buy-in for other projects focused on Automotive and Industrial
- Has their own PV devices but has been adding support for virtio over the past several years
- Has a group looking at functional safety

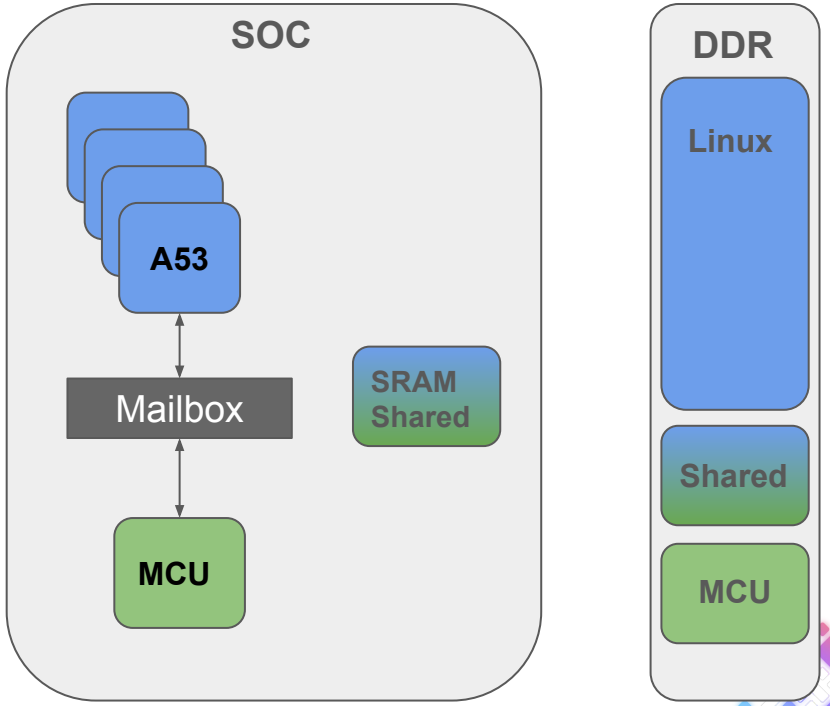
Zephyr

- A great reference for any RTOS
- An RTOS is important to show RT and functional safety
- Open Source & open governance
- Many CPUs and boards, Cortex-M, R, and A as well as many others. Tiny MCUs and big PCIe based systems. 32 and 64 bit.
- Easy to get started
    - (if you don't try to understand the cmake system or DT Macrobatics :)
- Has a group looking at functional safety

AMP Examples – AMP SOC

AMP SOC

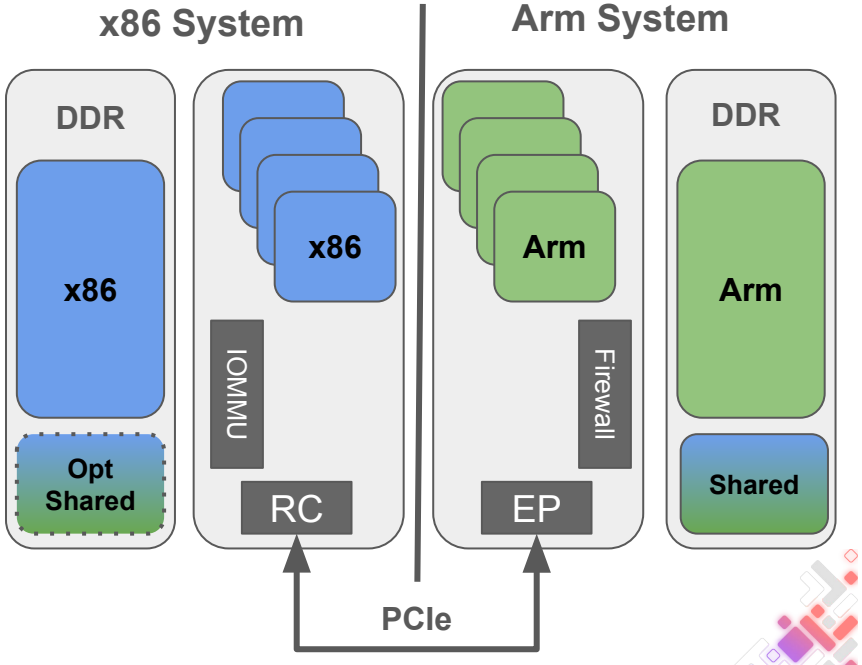
- A single SOC
- CPUs that are SMP Linux capable
- Other CPUs are MCU like, used for
  - Real time or IO offload
  - Safety or Security critical functions
  - Digital Signal Processing
  - Low Power standby w/ IO
- Examples:
  - ZynqMP: 4x A53s + 2x R5s [+FPGA]
  - STM32MP15: 2x A7s + 1 M4
  - NXP iMX8M+: 4x A53s + 1 M7 + DSP
  - TI TDA4VM: 2x A72s + 6x R5s + 3 DSPs
- This is classic OpenAMP



AMP Examples – AMP via PCIe

AMP via PCIe (and similar)

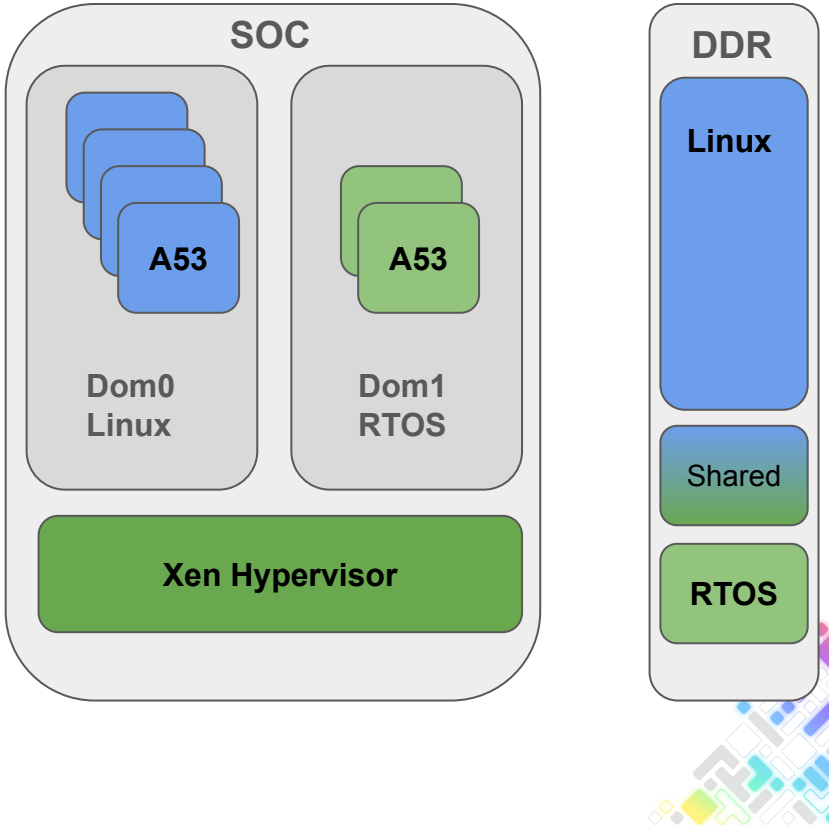
- X86 host with Arm SOC on a PCIe card
- Two PCIe RC systems connected with a non-transparent bridge
- UCIe and Chiplet ecosystem
  - Making these AMP systems more common and more customizable
- Two QEMUs using IVSHMEM
  - Good stand-in for the cases above
  - ~Model for a non-transparent bridge
  - Shared memory, MSI interrupts, and a doorbell MMR on each side



AMP Examples – Xen

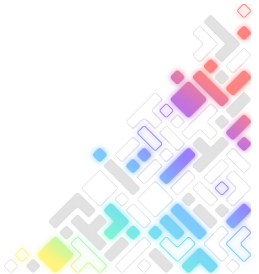
AMP via Xen

- Mixed Critical system
- Dom0less creation of critical RTOS domains at boot time
  - Real-time
  - Higher level of Functional safety
- Linux based DOM0 boots in parallel
- Other pCPUs and memory can be used by DOM0 to create non-critical DOMUs.



Why AMP Virtio for Xen

- Xen has its own PV devices
- Xen already does virtio-mmio and virtio-pci
- However, When doing virtio, Xen uses a IOREQ server
  - This request to the server hangs on to the vcpu that peeks and pokes the magic memory of the virtio transport area
  - This vcpu grab interferes with real-time and represents a week point in FFI from a functional safety POV
  - Using the virtio AMP methods, we can do virtio w/o the IOREQ server



AMP Virtio based on MMIO

Virtio MMIO Review

Virtio consists of the following

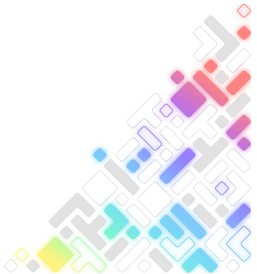
- Wire protocol of the various device types (blk,net,console,rng,etc)
- The bytes of the wire protocol content goes into buffers
- Device drivers use the Virtqueue API to put buffers into and take from virtqueues
- vrings are the implementation of the virtqueue API
- Transport level does:
  - Device type and status
  - Feature negotiation
  - Virtqueue configuration, startup and shutdown
  - A free form config data space

Virtio-MMIO consists of the following

- 256 bytes of MMR space
  - Magic, Version, & Device Type fields
  - driver & device feature bit 32 bits at a time
  - Virtqueue info, 1 virtqueue at a time
  - (v1.2) Per device shared memory segments, one segment at a time
- 256 bytes (commonly) of config space
- Relies on hypervisor intercepting each read and write to the mmio space
- Features, virtqueues and shm each use a register to specify which index is being addressed
- Notifications send via “MMR” write
- Notification receive via auto clear on read of another MMR

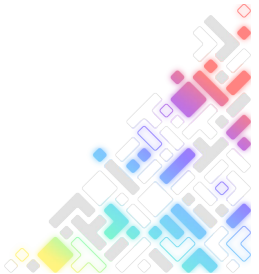
AMP Virtio

- Three actors: device, driver, and coordinator
- Coordinator is a role that can be simple or complex:
  - A boot loader filling an area with zeros or a binary file
  - Part of device side or even driver side
  - A VMM dynamically doing hot-plug / un-plug
- Always
  - AMP virtio memory area (512 bytes)
  - Use case specific bi-direction notifications
- Typical use
  - Single shared memory between peers
- Other uses
  - Multiple pre-shared memory areas (ex: one SOC SRAM, one DDR)
  - Device can access all of driver side memory (Xen DOMU or KVM host)
  - Driver dynamically grants access to memory via IOMMU or hypervisor facility



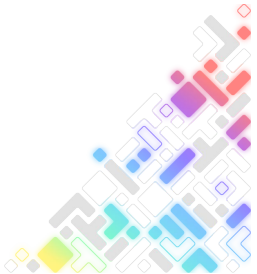
AMP Virtio Problems - Transport Level

- Initialization delay
  - Both sides are starting at the same time
  - wait for magic numbers and device type to be valid
- AMP mode recognition (driver side)
  - Use unique Magic/Version combo instead of that used by virtio-mmio magic
- Index based configuration
  - No hypervisor magic memory
  - Set index, send notification and await acknowledgement before using other MMRs
  - Send notification after data update
- Notification model
  - 2 Levels: unique and shared
  - unique uses two pair of single writer 64 bit bitmasks and xor
  - shared is platform specific and may be shared with other uses
  - **AMP SOC**: mailbox or cross core IRQs
  - **PCIe**: MSI[x] and Doorbell MMR
  - **Xen**: Xen events
- Config changes
  - Generation counter is odd means data is unstable



AMP Virtio Problems - Memory Level

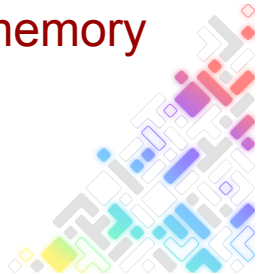
- **Admittedly this is the fuzziest section!**
  - We do have one prototype with Linux kernel as driver side
  - It needs more work
- Device side needs to know physical address of driver side
  - Rendezvous data structure in shared memory allows driver side to declare its PA for this memory segment
- Virtqueues and buffer allocation
  - Driver side OS changes needed
  - Device specific DMA space
  - SWIOTLB
- Memory coherency
  - Avoid cases where Linux side needs to do cache-ops
  - Use uncached memory (bad for DDR)
  - RTOS library has option to use cache-ops
  - Future: DMA subsystem and DMABUFs
- Not all uses have these issues
  - Device PA == Driver PA?
  - Device can r/w all of Driver side memory?
  - Driver can grant memory access to Device dynamically
  - Memory is always coherent



Status

Status: This work

- Not as far as I had hoped!
  - I had hopped to show many previous demos and prototypes rolled up and updated for this version of AMP virtio
- What we do have:
- White paper describing this version of AMP virtio
- My Demo:
  - Easy setup to run QEMU / Xen / Debian Dom0 and various DomUs
  - Zephyr hello world in Xen DomU created by xl and as dom0less
  - NOT YET: Dual Zephyr DomUs with event exchange w/ shared memory

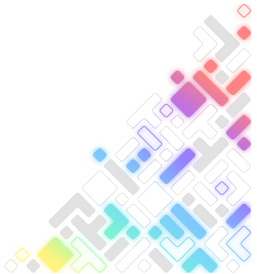


Status: Prior demos and prototypes

- Zephyr on A53 with virtio drivers to QEMU virtio-mmio devices
  - Zephyr using normal virtio-mmio
  - This is close to upstream ready
- Zephyr on ZynqMP R5 virtio mmio drivers to Linux user space devices
  - Previous version of AMP virtio-mmio using transport level prior knowledge on both sides
  - Uses modified kvmtool as user space device provider
- Linux on STM32MP157 A7 virtio-i2c to Zephyr on M4 I2C device
  - Previous version of AMP virtio-mmio with single signal between driver and device
- Dual QEMU w/ rpmsg
  - Shows virtques/vrings being used across PCIe on AMP system model

AMP Virtio a Change in Direction

- Today we are based on virtio-mmio
- Issues
  - Config space still has issues with simultaneous change from driver and device
  - Lots of round trips for index based config
  - Kernel feedback: “There is enough difference here we should just add a new transport provider and not modify existing virto-mmio”. So not a lot of benefit to aligning to mmio is achieved
- Probably change
  - Drop mmio alignment
  - Keep AMP notification model
  - Keep option for rendezvous data structures
  - Define a message passing structure in shared memory
  - Use messages like set-virtqueue(5, size, PA1, PA2, PA3. …)
  - Look to align messages with vhost-user or (maybe) virtio-ccw



Thanks and Credits

Thank you for your attention!

Credits & References:

Zephyr Virtio Native & HVL:
Danut.Milea@windriver.com

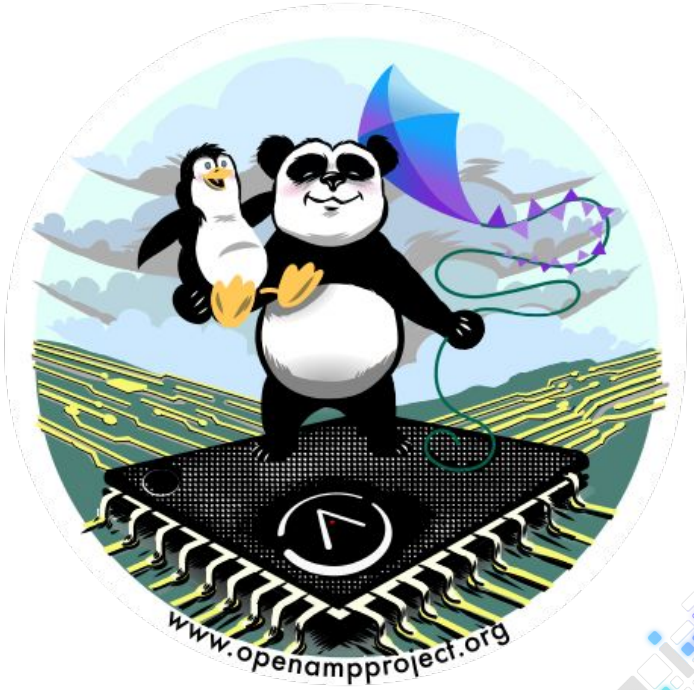
ST I2C demo:
arnaud.pouliquen@st.com and
intern Nicolas Granger

Dual QEMU w/ IVSHMEM RPMSG:
Felipe Neves

Zephyr event exchange program:
stefano.stabellini@amd.com

Help on Dom0less & RT:
michal.orzel@amd.com

Please ask me about our OpenAMP stickers



EMBEDDED LINUX CONFERENCE